Checkout and Launch Control System
(CLCS)
System Software Development Plan

Volume 1

Prepared By:

CLCS System Software

DP
Kennedy Space Center, FL 32899

CLCS System Software Development Plan

Original Signed By:

Retha Hart
CLCS Program Manager

Original Signed By:

Tom Swanson
NASA Safety & Mission Assurance

Original Signed By:

Larry Wilhelm
CLCS System Software

Original Signed By:

Kirk Lougheed
CLCS Systems Engineering

Concurrence:

Original Signed By:

Dennis Fougnie
CLCS System Subsystem Engineering

Original Signed By:

Ben Bryant
CLCS Applications Software

Original Signed By:

Ralph Esposito
USA CLCS Project Manager

Original Signed By:

Mark Dotterweich
LMSMS&S CLCS Project Manager

Original Signed By:

Jeff Wheeler
CLCS User Liaison

Original Signed By:

Henry Yu
Dynacs CLCS Project Manager

PREPARED BY: L. Wilhelm, DP-4

REVISION HISTORY

REV	DESCRIPTION	DATE

LIST OF EFFECTIVE PAGES				
Dates of issue of change pages are:				
Page No.	A or D*	Issue or Change No.	CR No.	Effective Date**

Appendicies:

Appendix A:

System Software Development Process

1. Introduction

1.1 Purpose

This document defines the System Software development plan to be used on the Checkout and Launch Control System (CLCS) project.

This document provides the guidelines for each CLCS software developer to accomplish the design, code, test and delivery of CLCS system software.

1.2 Referenced Documents

- CLCS Design Panel Assessment Template Document, 84K00XX -TBD
- CLCS Design Panel Software Requirements and Design Template, Document 84K000XX-TBD
- CLCS Programming Standards Document, 84K07500-010
- CLCS Systems Engineering Management Plan, Document 84K00053
- CLCS System Design Document (SDD), Document 84K0021
- CLCS Configuration Management Plan, Document 84K00052
- CLCS System Test Plan, Document 84K00056
- CLCS Program Management Plan, Document 84K00050
- CLCS Project Plan, Document 84K00051
- CLCS Project Safety and Mission Assurance Plan, Document 84K00055
- CLCS Transition Plan, Document 84K000XX-TBD
- CLCS Certification Plan, Document 84K000XX-TBD

1.3 Acronyms and Definitions

The latest information on CLCS acronyms and definitions can be found on the CLCS home Web page: http://lpsweb.ksc.nasa.gov/CLCS/

2. Scope

2.1 Volume I: System Software

Volume I of this document applies to all System Software to be delivered for the CLCS project. It includes Gateway software development, User Applications software development, SDC interface software development, re-used software and COTS software except where noted.

2.2 Volume II: Application Software

Volume II of this document applies to all Application Software to be delivered for the CLCS project. It includes all test and checkout application software for LCC area sites and remote sites such as CITE and HMF.

It is the goal of CLCS to employ a common development methodology for both System Software and Applications Software to the extent practical.

3. Applicable Documentation

3.1 Parent Documentation

This plan is derived from the CLCS System Engineering Management Plan (SEMP), which defines the overall system engineering process for the CLCS Project. Selected information from the SEMP is included in this document to aid in understanding of how the CLCS software development process fits into the overall system engineering process.

3.2 Software Programming Standards

The detailed software programming standards to be employed in CLCS are defined in the CLCS Programming Standards Document (84K07500-010).

3.3 Detailed Process Documents

The detailed processes to be used for CLCS software development are listed below. These processes will be made available to all CLCS developers and will be periodically updated as required.

- Unit Test Process (84K000XX-TBD)
- System Test Process (84K000XX-TBD)
- Issue Resolution Process (84K000XX-TBD)
- Risk Management Process (84K000XX-TBD)
- Software Re-Use Process (84K000XX-TBD)
- Code Walk-Through/Inspection Process (84K000XX-TBD)

3.4 Software Configuration Management Document

Software configuration management details are contained in the CLCS Configuration Management Plan, Document 84K00052.

3.5 System Test Document

System Software testing details are contained in the CLCS System Test Plan, Document 84K00056.

4. CLCS Project Overview

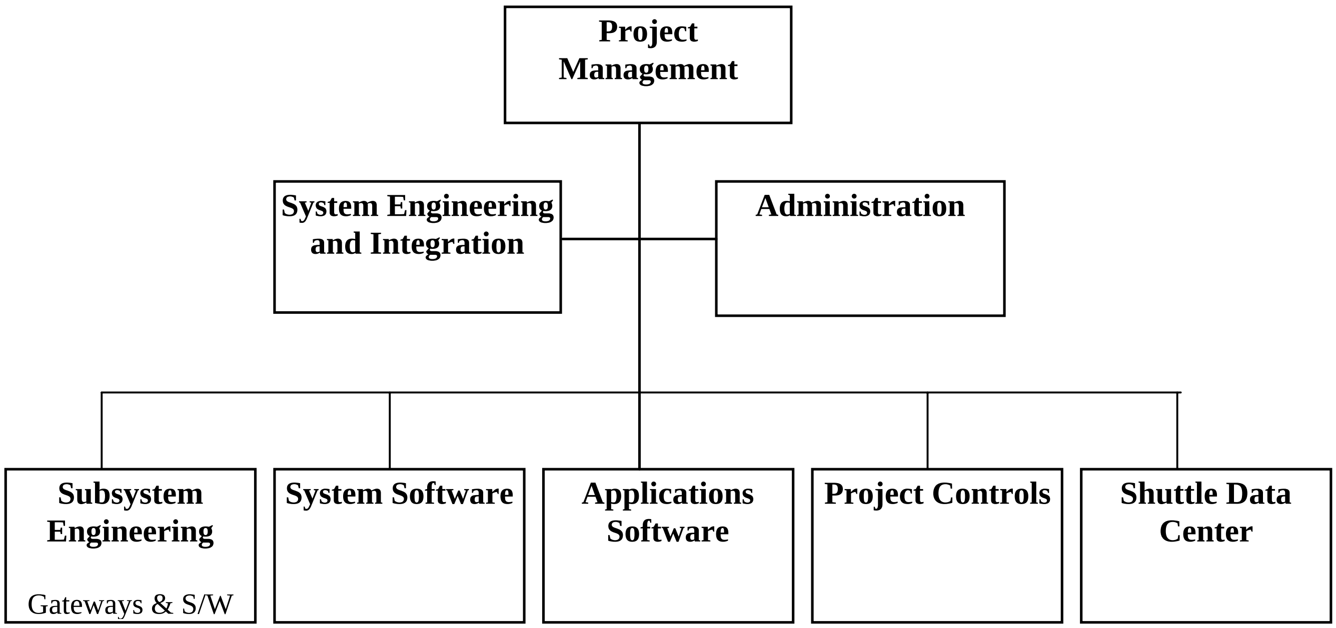
The CLCS will replace the existing Launch Processing System (LPS) used at the Kennedy Space Center (KSC). The LPS monitors and controls ground support and flight equipment during the testing, pre-launch, launch and On-Orbit phases of the Space Shuttle.

The CLCS is a five year project with ten planned software deliveries to incrementally provide CLCS capabilities. The software deliveries are planned at six month intervals, culminating in a fully operational CLCS system.

The overall CLCS project team is managed by NASA. Team member companies include: NASA, INET, Lockheed-Martin Space Mission Systems and Services (LMSMS&S), United Space Alliance (USA), Boeing, and EG&G. Each contractor provides management of their contractor teams under the overall CLCS Project management.

5. CLCS Organization

The CLCS Project will be managed according to the following abbreviated organizational chart.



5.1 System Engineering and Integration

Provides direct support to the CLCS NASA Project Managers. The System Engineering and Integration Division is responsible for the following:

- Project level strategic planning and coordination
- System level hardware, software, platform, and network architecture development
- System level requirements capture, reliability and maintainability analysis, safety assurance, security engineering, and pre-production configuration management
- System level integration and test certification plans, and the coordination of technology studies

5.2 Subsystem Engineering

Provides the design, procurement, implementation and delivery of all necessary CLCS hardware, gateway software and network management services.

- Procures or develops CLCS hardware
- Procures or develops Gateway hardware and software
- Procures or develops Network hardware and software
- Provides Network management and security
- Procures or develops Console hardware

5.3 System Software

Provides the design and development of all CLCS software designated as “System Software”.

- Defines the software development process
- Performs CSCI, CSC subsystem design
- Produces mid-level software design specifications
- Produces tested, configuration managed, documented and delivered System S/W code.

5.4 Applications Software

.Provides the design and development of all User applications Software.

- Reengineering of all Goal applications
- Produces tested, configuration managed, documented, and delivered Applications Set Code
- Provides a common repository of applications code

5.5 Project Controls

Provides and facilitates the overall CLCS project planning, change control, analysis and facility modification management.

- Management of the CLCS Change Control process
- Management of CLCS facility modifications
- Set activation
- Project Compliance and process definition
- CLCS planning, analysis and project performance measurement
- Administrative project support

5.6 Shuttle Data Center (SDC)

Provides the SDC hardware and software support to CLCS build, data bank, data recording, Informational services and SDC operations.

- Provides CLCS system and TCID build Products and services
- Provides DBSAFE data base and support
- Provides data recording services
- Provides configuration management of all developed and delivered products
- Provides SDC operating systems, development environments and tools
- Provides CLCS BASIS, document viewing and data distribution

6. Related CLCS Groups

CLCS software design and delivered code is produced under the oversight of the following groups under Systems Engineering and Integration. Each group operates under chartered responsibilities depicted in the System Engineering and Management Plan (84K00053). The group's functions are summarized below:

6.1 System Design Team (SDT)

The System Design Team (SDT) is responsible for the complete CLCS system level design architecture. System Design is the lead organization that coordinates and directs the CLCS System Design Team. This design responsibility includes coordinating the System Hardware Architecture, Software Architecture, Sub-system Design, Network Design, and all external interfaces to CLCS.

The System Design Team is responsible for:

- CLCS System Level Specification (SLS)
- CLCS System Design Document (SDD)
- CLCS External Interface Descriptions (IDD)

The System Design Team is also responsible for defining and documenting the overall CLCS System Software, Applications Software and Gateway Software architectures. The System Design Team works closely with CSCI leads, Systems Engineering and the CLCS User Community to ensure a software architecture that meets approved requirements.

The SDT will provide (as part of the overall software architecture):

- CLCS Software Architecture Documentation
- Software Architecture Issue Resolution
- Coordination of Interfaces between CSCI's

6.2 Hardware Architecture Team (HAT)

The Hardware Architecture Team (HAT) is responsible for defining and documenting the overall CLCS System Hardware Architecture. The team also works closely with HWCI and CSCI leads, Systems Engineering and the CLCS User community to ensure a hardware architecture that meets approved requirements.

The HAT is responsible for:

- System-level hardware architecture definition
- System-level network architecture definition
- Hardware issue resolution.

6.3 Engineering Review Panel (ERP)

The Engineering Review Panel (ERP) is chaired by the System Engineering Division Chief or designee.
The ERP resolves system design issues that are of major importance, such as safety issues, significant system design issues, system performance, or those issues that significantly involve both hardware and software.

The ERP is responsible for:

- System-Level issue resolution such as design, testing, performance or safety issues

6.4 CLCS System Integration and Test (I&T)

CLCS System Integration and Test (I&T) is responsible for the complete system level integration of delivered products and testing of all delivered products to insure that all system level specifications and product level specifications have been met. In addition, Integration and Test is responsible for providing direction and coordination of the system level configuration management policies and procedures, and the system set build policies and procedures.

System Integration and Test is responsible for:

- CLCS System Test Plan
- CLCS Configuration Management Plan
- CLCS Delivery Specific Test Plan and Procedures

6.5 CLCS Delivery Management

CLCS Delivery Management is responsible for providing support, on a delivery based focus, to the CLCS Project Managers. This support includes product development schedule tracking, procurement schedule tracking, integration tracking, facility modification schedule tracking, and final delivery verification and accountability.

The CLCS Delivery Management is responsible for:

- Scheduling and tracking of all deliverable hardware and software products
- Periodic status presentation of major delivery phases and milestones to CLCS developers and project management
- Providing a forum for delivery issue resolution and issue tracking
- On-going delivery support such as phasing, dependencies and delivery issue resolution

7. System and Software Documentation

7.1 System Level Specification (SLS)

The System Level Specification (SLS) contains the CLCS System Level requirements. The System Level Specification is derived from the following sources:

- Existing Launch Processing System Requirements
- LPS user requirements inputs
- Shuttle Upgrade Project Requirements
- External to LPS System Interface requirements (i.e. Ground Measurement System, HAZ GAS)
- KSC Resident CCMS, CDS, RPS, expertise

The Checkout and Launch Control System is defined by a set of capabilities needed to successfully launch a Space Shuttle (and later, future space vehicles) from KSC. These capabilities include monitoring and controlling ground support and flight equipment during the testing, pre launch and on-orbit phases of the Space Shuttle. These CLCS System Capabilities are defined in the System Level Specification.

The System Level Specification is a living document. It is under configuration control after the first project Architectural Baseline Review. Changes, corrections, and additions to the SLS are under configuration control through the CLCS Change Control Board.

The CLCS project development is broken up into several software and hardware deliveries. The requirements captured in the System Level Specification are allocated to the appropriate CSCIs and HWCIs as part of the design process. The delivered CSCIs are designed, tested, and accepted to the requirements stated in the SLS and to the lower-level product requirements stated in the Software Requirements and Design Documents (SRDD's) .

The system level and lower-level requirements are incrementally met with each delivery. CLCS software functionality will be delivered in a series of incremental deliveries over 5 years. Each delivery will add increasing CLCS capability in functioning segments, or "threads", or non-thread products. Each incremental delivery will be tested and delivered to CLCS Users. Both Developers and Users will provide feedback on delivered software quality and functionality.

The CLCS system capabilities to be developed for each incremental delivery are defined in the Delivery Document.

SLS requirements correspond to IEEE-2167 Specification "Level-A" requirements. SRDD requirements, described in the following paragraphs, correspond to the IEEE-2167 "Level B" and "Level C" requirements.

7.2 System Design Document (SDD)

The CLCS System Design Document (SDD) (84K0510-XXX TBD) contains the definition of the CLCS hardware and software architecture including an overview of each software CSCI. The System Software Requirements and Design Document (SRDD) continues from that point to provide a complete picture of the software design, both for the development and for the sustaining phases of CLCS.

7.3 Software Requirements and Design Documents (SRDD)

The Software Requirements and Design Documents (SRDD's) are the primary software product specifications that document the design of each delivered CSCI and CSC.

The SRDD's are a collection of CSCI and CSC-Level design documents which contain both the CSC software design, detailed design and the requirements that the CSC is meeting. Each SRDD document structure must conform to a Template defined in the CLCS Design Panel Software Requirements and Design Template, (document 84K000XX-TBD).

The Software Requirements and Design Document (SRDD):

- Is produced by the CLCS Software Developer
- Documents the CLCS software requirements, design and detailed design
- Is organized by CSCI and CSC
- Incorporates templates for consistency
- Is a living document, under periodic revision
- Is released as a new revision with each CLCS software release
- Adds increased depth with each Design Panel presentation
- Applies to all System, Gateway, CLCS-SDC and Applications Software
- Is numbered at the CSC and CSCI level
- Will be maintained for the life of CLCS
- Will be stored in a NASA/KSC document repository

The CSC SRDD documents are individually numbered. Each CSC SRDD document is referenced by a top-Level CSCI document which is also numbered. Each SRDD document (CSC or CSCI) contained in a CLCS software release will be revised, if necessary, for that release and approved by the Design Panel process, be given an updated revision number and released to the User.

The CSCI-level document provides an overview of each CSCI's functionality. The CSCI document references the CSC's it contains. Each CSC document contains the actual code design and mid-level requirements per the Design Panel Template.

7.4 Software Documentation Development Schedule

The following Matrix provides an overview of the required software documentation and depicts when a specific document is required in the software development process.

Software Documentation Development Schedule
(Figure 1)

Document Name	Description	Date Due Prelim	Updates	Date Due Final	Comments
CSCI Overview Specification	Provides an overview of the CSCI and a brief description of each CSC within the CSCI. This document is a forward for the CSC Reqs & Design Spec.	Reqs Panel		CIT Complete + 10 Days	N/A for Application S/W
CSC Development / Implementation Schedule	A schedule of the implementation tasks, dependencies, and significant milestones planned for providing the CSC in a delivery.	Reqs Panel	Updated Weekly	N/A	Generated in MS Project and reported to Delivery Manager on weekly basis.
CSC Requirements and Design Specification	Contains the functional and performance requirements and the detailed design for the CSC for the current and prior deliveries.	Reqs Panel	Detailed Design Panel	As-Built at CIT + 15 Days	Apps S/W produces a separate requirements and design spec at the CSCI level
CSC Interface Definition Document (IDD)	Contains the requirements and design for establishing an interface with another CSC. The IDD represents an interface agreement that must be signed by all parties involved in the I/F.	Reqs Panel + 10 Days	N/A	Detailed Design Panel + 10 Days	Done on a CSCI basis for Apps S/W
CSC Application Program Interface (API) Manual	Contains the description of any programmatic interfaces provided by the CSC. The description includes usage, list of arguments, return values, error codes, and any special instructions. Document formats follow UNIX Man Page for on-line utilization.	Reqs Panel + 10 Days	N/A	Detailed Design Panel + 10 Days	N/A for Application S/W
CSC Test Plan	Defines the test environment and test cases to be executed to verify the functional and performance requirements for the CSC.	Reqs Panel	N/A	Detailed Design Panel + 10 Days	This is the CSCI SVP for Application S/W (see below).

Software Documentation Development Schedule (continued)
(Figure 1)

Document Name	Description	Date Due Prelim	Updates	Date Due Final	Comments
CSC Test Procedures	Defines the detailed procedures to be executed for each test case for the CSC. These procedures include test setup, test tool descriptions, procedure steps, and expected results.	CIT- 15 Days	As Required	CIT Start - 3 Days	Contained in the CSCI SVP for Apps S/W.
CSC Test Report	Documents the results of the formal CIT for this CSC. This report includes the "as run" test procedures, test data captured/analyzed, and a description of any anomalies (issues) written during the test.	N/A	N/A	CIT + 10 Days	Done on a CSCI basis for Apps S/W
CSC User Guide	Contains an overview of the functionality provided by the CSC and the detailed instructions on how to operate or invoke the various functions. Includes "As built" displays, user interactions required, and informational or error message explanations.	Detailed Design Panel	N/A	CIT	
CSC Operational Notes and Installation Instructions	Defines any special operational notes on installing, configuring or executing the CSC that would not be covered in the CSC User Guide.	CIT - 15 Days	N/A	CIT	
Software Metrics Report	Provides a report on the development and test metrics for each CSCI/CSC as required by the CLCS Project Software Development Plan.	Reqs Panel	Monthly	N/A	This is done on an IPT basis for Apps S/W.
CSCI S/W Ver. Plan (SVP)	Provides the plan for the users to validate the application software and the underlying system.				N/A for System Software.

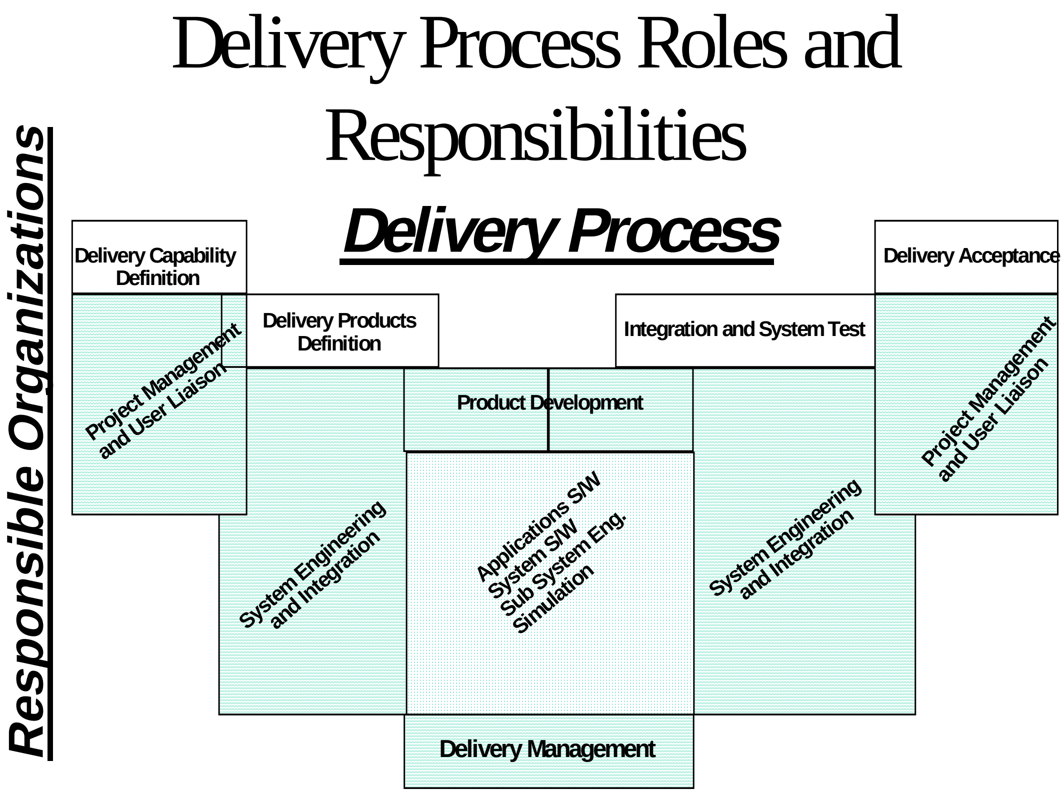
8. Software Design Process (overview)

The software design process is contained within the system design process and is divided into three distinct phases: Concept Design Panel, the Requirements Design Panel, and the Detailed Design Panel. Each Panel phase concentrates on specific products and activities in the delivery of both CLCS hardware and software.

The Software development process continues to occur after the third (final) Software Design Panel. In this phase, final detailed software design, code inspection, code walk through and integration tests are performed Products generated in the final phase are not reviewed in the formal Design Panels.

Note: The Concept Design Panel, the Requirements Design Panel, and the Detailed Design Panel are intended to incrementally provide the same level of review that the classical Preliminary Design Review and the Critical Design Review would as outlined in the outdated MIL-STD-2167. This method reduces user acceptance risk and provides a method of early detection of design problems.

9. Software Lifecycle Roles and Responsibilities



10. Software Delivery Definition

A strategic system engineering analysis will be performed in advance of delivery start. Each delivery is mapped out to meet specific and measurable project goals, providing capabilities to the user community at the earliest possible time and constraining the delivery to an achievable work content.

The delivery definition phase begins with a written agreement between CLCS Project Management and the User Liaison office which documents the specific capability of a software delivery. This includes:

1. Capabilities Provided For Operational Support
2. Facility Considerations
3. Affected External Organizations

The Five Year Project Schedule and Project Effectivity and Milestone chart, as well as the Strategic Engineering function discussed earlier will be used to facilitate this definition agreement activity.

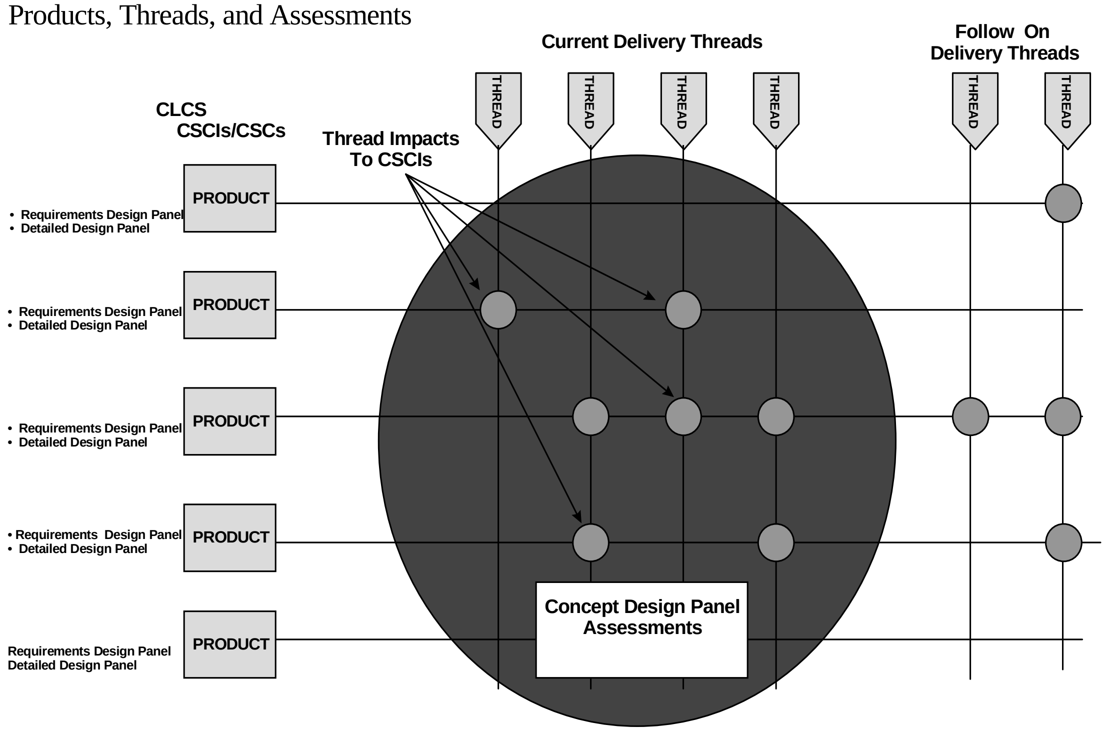
10.1 Threads

A thread is a specific system-wide capability defined and provided in a CLCS software delivery. It is a major system wide capability that can be exercised by designer and user as soon as it is delivered. It acts as a system test case during integration testing and usually involves many CSCI's and hardware platforms. It is defined, scheduled, tracked and tested. A thread lead is assigned early in the development process to ensure adequate design and oversight.

System Engineering and Integration use the concept of a system "thread" to capture a set of capabilities for each delivery. Threads encompass a system wide view of all software products (CSCI's/CSC's) and hardware products (HWCIs) that the system capability relies on. The diagram on the following page illustrates the product and thread relationship.

As the threads are defined for a specific delivery, they 'impact' CSCI's, CSC's and HWCI's. The intersection of the vertical lines (delivery threads) with the horizontal lines (CSCI's/CSC's/HWCI's) represent the assessment of the work required to provide the capabilities of a delivery.

Products, Threads, and Assessments



10.2 Thread Definition Activity

Thread definition is the initial activity for a delivery. The thread definitions are made by Systems Engineering and Integration. The thread definition includes participation from the User Liaison throughout the process. This process captures the required capability that the CLCS CSCI’s/CSC’s/HWCI’s must provide. The product of this definition phase is the Delivery Definition Document.

10.3 Delivered Products

CLCS delivered products are organized in general categories:

1. Thread Based Products
2. Non-thread Based Products

10.3.1 Thread-Based Products

Thread-Based Products are software products to be delivered that have been developed in accordance with allocated system level specifications to meet the intent of the thread defined capability. These products are described by the thread and are a configuration managed deliverable in the form of a CSC or CSCI.

A thread is not a delivered product. A thread describes a set of capabilities of associated CSCI's/CSC's/HWCI's and provides a 'system capability'.

10.3.2 Non-Thread-Based products

Non-Thread-Based products are those items that include, but are not limited to, the following:

1. Prototype Products, prototype software (e.g. Console Enclosures, COTS product evaluation)
2. Trade Study outcomes
3. CSCI development that is in direct support of a capability that is described by a later delivery thread
4. System software CSCIs such as operating systems, design tools, etc.

While these types of delivered products are not necessarily categorized by a thread, they are included in the project work breakdown structure and are tracked and managed accordingly.

10.3.3 Pathfinders

The purpose of a pathfinder is to produce, and usually demonstrate, the framework for a future design solution where the exact design direction is not clear or agreed-upon.

When required, a pathfinder will be created during the delivery definition and tracked with less formality than a thread. A pathfinder usually follows a shortened design process, and may include only the first or second design panel. It can be employed for either software or hardware designs.

In general, Pathfinders have the following characteristics:

- Lightweight in design process and schedule
- Prototype designs are produced and demonstrated
- User review is included in the design and in demonstrations
- Pathfinders use the Design Panel and Thread definition process (although abbreviated)
- Prototype (i.e. potentially through-away) code is produced
- Pathfinder code may be incorporated in a delivery but only after the requirements of this software development plan have been met.

10.4 Delivery Definition Document

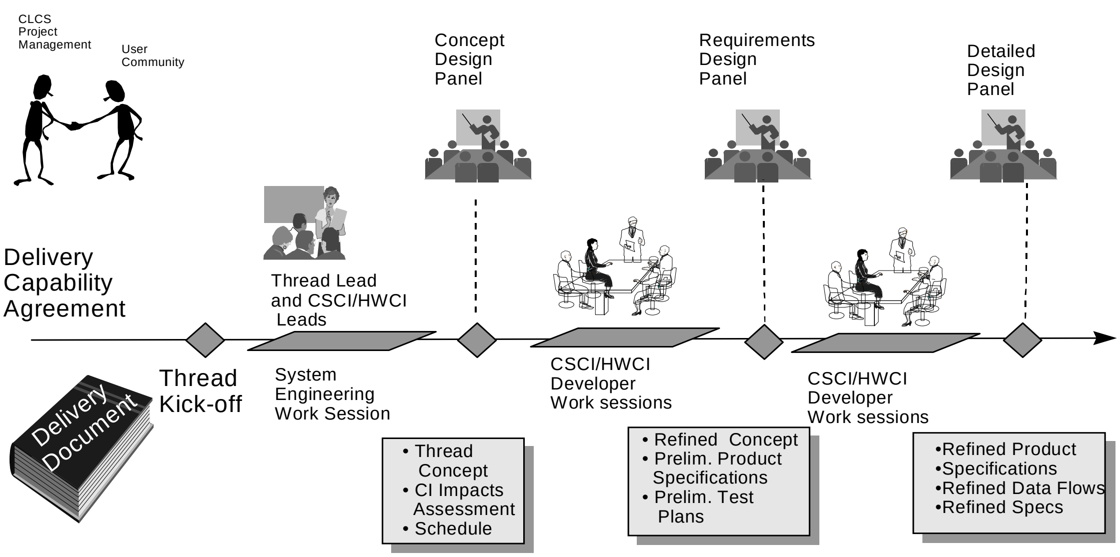
The CLCS Delivery Document describes all of the software threads and products that make up a delivery, both software and hardware. It is produced by the CLCS design team at the start of the delivery definition and evolves as the thread assessments solidify.

It is a living document until the successful completion of the Concept Design Panels. It then becomes the official agreement between Project Management, the User Liaison, System Engineering and Integration and the development organizations defining the work that will be performed for an incremental delivery.

Once the delivery document has been released, the project delivery manager begins to build the delivery schedule and track all delivered products.

11. CLCS Design Panel

CLCS Design Panel Process



11.1 Design Panel Process

The software development process is significantly linked to the CLCS Design Panel. All newly developed, non-COTS software, except as noted in Volume II for User

Applications software or pre-existing SDC software will be developed using the design panel process.

The Design Panel is a coordination activity rather than a detailed design activity that reviews the software design to the CSC level. Users are invited to every stage of the Design Panel.

The panel considers SLS requirements, current LPS system functionality, User's needs, and proposed operational concepts. Software is reviewed against the total system architecture. The review considers the total software life cycle including the sustaining phase.

The Design Panel process can be shortened in real-time. Panels can be combined or eliminated, depending on complexity or criticality at the discretion of the Panel Chairman. Special topics, Pathfinders or concepts can also be presented as panel agenda items.

11.2 Responsibility and Authority

The CLCS Design Panel reviews all hardware and software that is directly produced for CLCS. The review occurs incrementally in the concept, requirements, and detailed design phases. It occurs prior to testing and delivery of CLCS products.

The CLCS design panel process is the responsibility of the System Engineering and Integration Division. The lead of the SE&I Division is by definition the Design Panel Chairman. The Design Panel Chairman has designated Design Panel Co-Chairmen. In addition, a Design Panel Secretary is designated.

The Design Panel Chairman has the responsibility to coordinate the internal design panel meetings and provide/publish the agenda for the official design panels. The chairman has the responsibility to conduct the design panels, direct any actions to be worked, and insure that configuration control of the design panel products is provided.

The Design Panel Secretary is required to take notes during the official design panel meetings and track any actions. The actions captured in a design panel will be recorded and tracked in the Razor Issue data base. In addition, the Design Panel Secretary will provide an archive (hard copy) of each design panel, along with an attendance list of the panel participants.

11.3 Thread Leads

The primary responsibility of a thread lead is to provide the system level engineering expertise for the thread assessment phase of a CLCS delivery. The thread lead will coordinate and capture CSCI and HWCI assessment impacts that will be documented using the thread assessment template.

The thread lead is responsible for providing delivery dependencies to the Delivery Manager using the CLCS Dependency Form. These dependencies will be incorporated into the Delivery Manager's Production Report.

The thread lead's job continues after the Concept Design Panel. After the successful completion of the thread assessment the thread lead will monitor and provide system level design guidance for the development of the individual CSCIs and HWCIs that were impacted by the capability provided by a particular thread. The thread lead will provide system engineering expertise throughout the CSCI and HWCI development cycle when an issue of requirements and/or capability requires resolution. In addition, the thread lead will provide system level engineering expertise to the Integration and Test group from the time that the assessment is complete.

11.4 Concept Design Panel

The Concept Design Panel is the first review that takes place for a CLCS delivery or change to a previous delivery.

Concept panel data is presented by a member of the System Engineering and Integration team (or designee). This presenter is considered the "Thread Lead" for a particular thread for the next delivery. The purpose of the Concept Design Panel is to provide the concept and an assessment of work required to meet all delivery capabilities.

Tasks to be completed prior to the Concept Panel:

1. Completion of the Thread Assessment Template including:
    - SLS Requirements mapping to threads
    - Draft of the development schedule
    - Description of tasks (e.g. working groups, ops. concept etc.)
    - Concept overview
    - Statement of work analysis and refinement
    - H/W and S/W diagrams
    - Labor assessment summary in labor months
    - Any delivery document updates or corrections
    - Dependencies
    - Items to be procured
    - Test and training requirements

- Issues
- Design Risk Assessment

2. Coordination of interface requirements and functionality with affected CSCI and HWCI leads
3. Creation of input for the Requirements Design Panel
4. Understanding of major concept of operations with the Users and CSCI leads

Data to be presented at the Concept Panel:

1. Completion of the Thread Assessment Template (reference 84K000XX TBD)
2. High Level System Overview of the Thread Capability
3. System Level Requirements satisfied by the delivery of the capability described by the thread.
4. Assessments in labor-months to complete the work for the delivery

11.5 Requirements Design Panel

The purpose of the Requirements Design Panel is to provide a review of the CSCI/CSC requirements and preliminary design. The material for the Requirements Design Panel is presented by the software CSCI Lead (or designee).

Tasks to be completed prior to the Requirements Panel:

1. Completion of the Software Requirements Design Panel Template (Document 84K000XX-TBD) including:
    - Groundrules
    - Functional requirements (level B and C)
    - Preliminary Design (overall software structure or class diagrams)
    - Performance requirements
    - Interfaces, data Flow
    - Test planning
    - Issues

2. Coordinate interface requirements and overall functionality with the Thread Leads

Data to be presented at the Requirements Panel:

1. Present the completed Requirements Design Panel Template
2. Present CSCI Preliminary Design information
    - First cut of class/object model or structure diagram
    - Specific “use cases” or scenarios generated from requirements

- First cut of process-level diagrams and process interaction diagrams

3. Provide input into the Detailed Design Panel Template

Post Requirements Panel:

- Resolve open issues

11.6 Detailed Design Panel

The purpose of the Detailed Design Panel is to provide a final review of the work to be performed prior to code generation. The design is reviewed at the top-level for good design practice. The panel ensures the design meets requirements and defined concept of operations. The material is presented by a CSCI Lead or delegated presenter.

In some cases, coding may be authorized to start prior to completion of the Detailed Design Panel. This authorization is by the Design Panel Chairman.

Tasks to be completed prior to the Detailed Design Panel:

1. Completion of the Detailed Design Panel Template (Document 84K000XX-TBD), refinement of all Concept Panel data, and refinement of object models.

   (note: not all are items are required, depending on use of a Structured or Object Oriented design process)

   - Requirements update (if any)
   - Groundrule update (if any)
   - Detailed Design Diagram
   - External Interface Diagram
   - Detailed Data Flow Diagram (structured)
   - Object Class Diagram (OOD)
   - State Diagrams (OOD)
   - System Messages
   - User or System Displays
   - API definition
   - Simulation Models
   - Detailed dependencies
   - Detailed Data interfaces
   - Syntax Diagrams
   - Input data and formats
   - Recorded data
   - Table formats

- External Interface test plan
- Documentation described in the Software Documentation Development Schedule (figure 1)
- Issues

Data to be presented at the Detailed Design Panel:

- The completed Detailed Design Panel Template

Post Requirements Panel:

- Resolve open issues

12. Coding

After completion of the Detailed Design Panel, software coding is completed per the CLCS Programming Standards Document, 84K07500-010. Final documentation updates are made per the Software Documentation Development Schedule (figure 1).

No further panel presentations are necessary and the developed code proceeds to Unit Test then CSCI Integrated Testing (CIT) and System Test.. Developers support software testing through software delivery completion. Post software delivery problem reports (Razor issues) are tracked in the CM database.

12.1 Code Inspections and Walk-Throughs

The code inspection and walk-through process can be combined into a single process or can be performed individually. Both must be performed on all custom software developed for CLCS.

The purpose of code inspections is to ensure that the guidelines of the CLCS Programming Standards Document have been followed. The purpose of the walk-through is to provide an independent look to verify that program function, logic, structure, data and methods meet good design practice and with minimum complexity.

It is the responsibility of the CSCI lead to perform and document both code inspections and walk-through for each delivered CSCI. Code inspections do not have to be combined with the walk-through and can be done on an informal basis by a peer or CSCI lead with the results being recorded.

Documentation recording the inspection/walk through reviewers, date and results will be retained in the Razor CM tool. The name of the actual programmer is not required to be retained. However, the CSCI/CSC name is required.

In general, code inspections will have the following characteristics:

- Notice of a review will be given three days in advance
- Two or three reviewers present, including from outside the current CSCI
- Reviewers will review for standards followed, technical content, good coding and design processes

12.2 Code Error Detection Tool

In addition to the testing processes stated in this document, it is planned that CLCS code will be analyzed for errors by an automated method under the auspices of the IV&V group. Current plans include utilizing the McCabe tool set to accomplish this task.

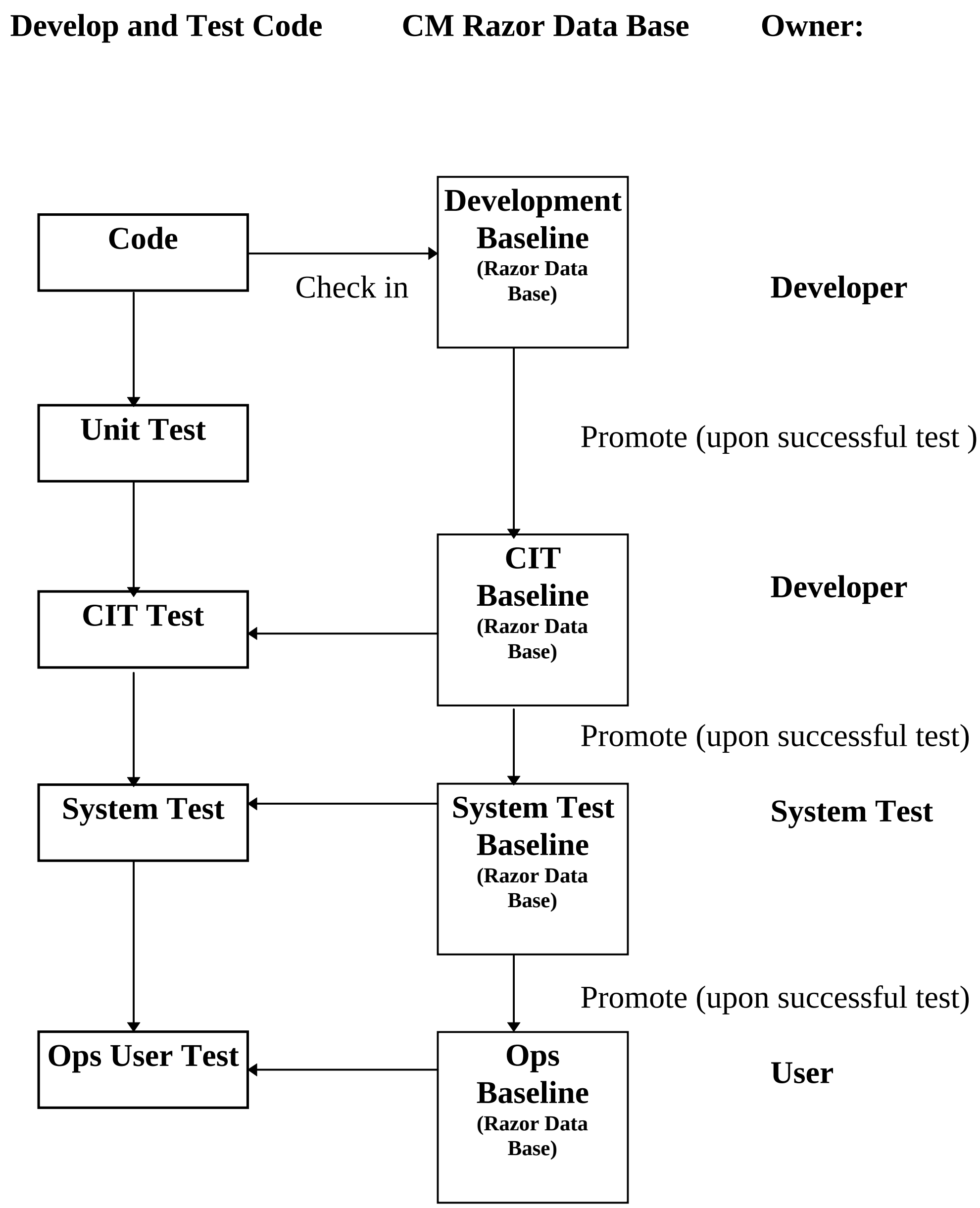
12.3 Software Development Folders

TBD

12.4 Code Configuration Management (CM)

Through the code development and test lifecycle the software code undergoes an increasing level of configuration management as it is coded, tested, and released.
The Razor (COTS) configuration management data base contains the developed code in all stages of production. Code is checked into Razor after initial coding and is "promoted" to increasing levels of CM as it successfully passes each phase of testing. The following flow describes the CM promotion process:

The Razor data base contains the developed code in all stages of production. Code is checked in to Razor after initial coding and is "promoted" to increasing levels of CM as it successfully passes each phase of testing. The following diagram describes the CM promotion process:



Software CM Promotion Process

13. Software Testing

Software testing consists of Unit Tests performed by the CSC developer, Unit Integration Testing, performed by the CSC developer; CSCI Integration Tests (CIT) performed by the CSC developer in conjunction with a Quality representative and the Systems Test group; and System Test performed by the Systems Test group in conjunction with a Quality representative.

13.1 Unit Testing

Unit Testing is the responsibility of the software developer. Unit testing is performed on software components to check functionality and performance in a standalone mode or in a debug environment. Unit Testing is to debug the software to minimize functional and integration errors.

Quality Assurance monitoring is not required. However, problems encountered during Unit Testing are recorded in the Razor data base and tracked, awaiting closure by the CSC developer and verification of closure by the CSCI lead. Closure is normally accomplished by additional unit testing unless the CSCI lead verifies, with the concurrence of Systems Test and Integration and NASA Quality that the problem can adequately be closed by other methods such as analysis or inspection.

- Unit Tests will be developed to perform functional testing by the CSC developer
- Test cases will check both normal and error conditions for each input parameter
- Unit Test cases will be maintained in the Configuration Management Repository for future regression testing
- For each code change, the associated Unit Test case will be modified as necessary and then performed
- Test results will be maintained in the Common CM Repository to provide for comparisons of follow-on tests
- All newly developed software, regardless of criticality, will undergo Unit Testing

13.2 Unit Integration Test (UIT)

Unit Integration Test (UIT) is performed by the software developer to verify basic functionality and successful integration of a set of programs (CSC's). The UIT is tested against Functional Requirements and can be the first level of testing for reuse software. UIT's are witnessed and signed-off by the CSCI lead and QA. The number of tests and test cases for UIT's are finalized at the Detailed Design Panel. UIT's are performed in the development environment.

Once Unit testing is successfully complete, code is promoted in Razor to the UIT baseline. The code is not yet under formal control and can be modified by the CSC developer. on.

13.3 CSCI Integration Test (CIT)

CSCI Integration Testing (CIT) is a test performed on software to test functionality and performance in an integrated environment against other software components within a hardware testbed. The purpose of CSCI Integration Testing is to ensure that the software has no known functional and performance problems prior to entering the formal System Test.

The CIT environment approximates the actual real-time operational environment as closely as possible. A representative suite of application software may be loaded as well as any other interfacing software. Either development tested software or simulation software may be used. Integrated testing may be also be performed against a hardware test bed (e.g. Sail, KATS). Integrated testing is the responsibility of the software developer. User participation is encouraged on a non-interference basis.

After successful completion of the CIT the software is placed under formal CM control.

Formal CM control includes:

- NASA-Quality verification of fixes
- Formal closure of Razor issues
- Code must pass the previous test before it can be checked into the current test baseline

13.4 System Test

The system level testing is completed against the System Requirements (SLS) and Design Documents (SRDD's). System Level Testing of a software delivery is organized by threads.

After successful CSCI integration testing dry-runs by the CSC developer, CM releases the CSCIs to the Systems Test group and the code is promoted to the System Test Baseline in Razor. Any fixes or new code is required to pass the previous test (or selected special testing) prior to checking in to the System Test Baseline. All code continues under formal control.

The System Test group verifies System Level Requirements, a representative set of mid-level or detailed system requirements and that external interfaces meet specifications. System functionality, performance, and system-wide data flow are tested.

After successful completion of Systems Test, software is released to CLCS users, who perform User-level Validation and Certification tests.

13.5 Regression Test

Representative subsets of functional and performance tests will be retained under CM control in the CM Repository for regression testing. The subset will be exercised with each release of CLCS software to the User, as a minimum and other selected times to prove functionality or performance trends.

Results of the regression tests will be recorded and retained for the life of the CLCS project in the CM repository. Results will be summarized and reviewed with CLCS project management, Systems Engineering and System Software on a release-by-release basis.

13.6 Software Issues

Software issues (software problem reports) can be opened by the developer anywhere in the development cycle and retained in the CM repository under Razor.

It is a requirement, however, that any issues that occur after CIT completion be recorded in the CM Repository. After CIT, issues are required to be closed in a formal manner (i.e. QA witness, re-run of the preceding test).

13.7 Post-Production Support

Software Developers provide post-production support for delivered products. Typically, post-production support includes:

- Systems integration support
    - Reviewing and commenting on documents
    - Helping write and refine procedures
- System testing assistance
    - Review and comment on system test Plan
    - Review and comment on system test Procedures
    - Support Dry Run system test
- Problem resolution

- User testing support
  Exact support is on an “as requested” basis

14. Independent Verification and Validation Support

CLCS Software will be independently evaluated by NASA and contractor representatives from the NASA IV&V facility at Fairmont, West Virginia. The CLCS project entered into formal agreement with the IV&V facility in June of 1997 to accomplish this task.

The CLCS project will be reviewed, including CLCS software, subsystems, documentation, and processes. Critical areas of software will be identified and evaluated using a Criticality and Risk Assessment (CARA) analysis.

The IV&V effort will provide an additional level of assurance as well as reduce systems engineering , integration, and operational use risks.

IV&V personnel report independently to the IV&V Facility NASA management and to the CLCS Project Manager. System Software personnel provide day-to-day IV&V assistance and information.

15. CLCS User Software Testing

15.1 User Software Validation

Software Validation is the formal process performed on software to test functionality and performance in an integrated environment against simulation software and in a configuration controlled hardware CLCS set. This post-CLCS development activity is performed on software released to the CLCS user. It is not to be confused with independent IV&V performed by the NASA IV&V facility at Fairmont, WV.

Validation may be performed against a hardware test and is the responsibility of system knowledgeable personnel with assistance from software production as required. Quality is required to support validation for changes to any critical components.

15.2 User Acceptance

User Acceptance is a systematic approach by which the user community gains the experience and builds confidence in the developed CLCS software. User Acceptance culminates in the use of software for vehicle processing. In the development phases of CLCS, user acceptance criteria will be identified by both the IPT and by external sources. External sources include senior shuttle technical representatives who will formally identify user acceptance requirements to the CLCS project.

16. Software Development Tools

16.1 Design Tool

Paradigm Plus has been chosen as the object-oriented design tool. This tool will be made available to each developer. Design tool environment files will be checked into Software Configuration Management along with the code, so that the programmer's desktops are also preserved.

The Rumbaugh method has been chosen as the Object Oriented design methodology and is supported by Paradigm Plus.

In cases where Paradigm Plus does not meet specific structured design requirements, other tools may be employed on a limited basis with the approval of System Engineering and System Software.

16.2 Drawing Tool

Visiotech has been selected as the CLCS drawing tool standard for CLCS drawings. Visiotech will be available to each CLCS team member. In cases where the drawing tool does not meet specific drawing requirements, other tools may be employed on a limited basis with the approval of System Engineering.

16.3 Office Support Tool Suite

Microsoft Office Tools (Word, Project, Excel, Exchange) are the standard project administrative and documentation tools. Each programmer will have these tools available. All CLCS project documents are to be generated using these tools.

16.4 Software Configuration Management

Razor, a project-wide (KSC, JSC, SDC) software configuration management tool will be used for managing all software development files, including development environment files. RAZOR will be available to all software development personnel. Complete information on the use of RAZOR is available via the CLCS Web home page.

16.5 Tool Updates and Future Automation

New Software development tools, or modifications to existing tools, are continuously being evaluated for use in CLCS. Tool changes will be extremely selective and must demonstrate a clear labor, cost or safety benefit before being incorporated in CLCS. This plan will be updated as new tool or tool capability changes occur.

Automation candidate topics currently being considered are:

- Requirements trace from the SLS to the SRDD
- Requirements trace to Thread Definition
- Tool assistance in documenting and putting on-line the CLCS software Architecture
- Creation of a CLCS Interface Document
- Use of Doc ++ to assist in documenting developed code

17. Software Development Environments

Several software development environments will exist to develop CLCS software. All share the same development. The standards and requirements set forth in this document apply to all custom developed software regardless of the development environment used.

17.1 Software Development Environment (SDE)

The Software Development Environment (SDE) hardware sets are used by programmers for developing and testing CLCS software. This environment includes all the hardware, tools, and software necessary to create, develop, test, manage, and document code. The primary CLCS Software Development Environments are located at KSC and at Lockheed-Martin Space Missions Systems and Services in Clear Lake, Texas.

17.2 Integrated Development Environment (IDE)

The Integrated Development Environment (IDE) is an operational environment subset, replicating the operational sets in configuration and performance, but not in total quantity of equipment. The IDE will be used to test Application Software in a verified operational configuration.

17.3 Development Set Locations

The IDE is located in the Launch Control Complex (2R23, 24 and 25). SDE’s 1 and 2 are located in the PCC room 3015. Lockheed-Martin operated systems are in contractor facilities in Houston (1322 Space Park Drive, Houston, TX 77058).

17.4 Development Environment Administration

The IDE and SDE configurations are controlled by the Integration Control Board (ICB) as well Lab Manager(s). Access to each site is controlled. Server backups are performed periodically. New accounts and account changes are performed by the CLCS System Administrators.

18. Software Change Control

New requirements to be incorporated into a baselined release (i.e. released to user) are controlled by the CLCS Change Control Board as are changes to baselined schedules.

Changes resulting from Software problems that do not reflect a new requirement are under the scope of this plan.

The policies and procedures that make up the change control process for CLCS software are discussed in detail in the CLCS Control Board Charter (84K00006) and the CLCS Configuration Management Plan (84K00027).

The process includes:

- Configuration Identification - selecting those items that will be placed under configuration control and when they will be controlled
- Change Control including the following products and procedures
    - CLCS Development Tools
    - CLCS Master Document Library
    - CLCS Software Library
    - Released/Delivered Software
    - Change Requests
    - Engineering Support Requests
    - Requirements changes and additions
    - Test Plans

19. Software Problem Reporting

Software Problem reports or Razor “issues” are tracked as defined in the Software Configuration Management Process.

The Web-based, COTS tool “Razor” implements this process. Detailed instructions are depicted on the Razor Web CM page: http://www-de.ksc.nasa.gov/clcs/cm/

20. Software Product Assurance

The Software Product Assurance Program is to assure the quality of all software and its documentation.

CLCS incorporates full time NASA, USA, LMSMS&S and other S&MA personnel to ensure quality of the delivered software product. Their duties include:

- Review and sign-off at the CIT test phase and beyond for all subsequent software tests
- Monitoring and sign-off of the CIT Test Procedures
- Monitoring and sign-off of the System Test Procedure

21. Software Security

All System Engineering and Development Organizations will identify security-critical CSCI's or CSC's whose failure could lead to a breach of NASA security.

CLCS System Engineering and Integration will develop a security plan to assure that the requirements, design, implementation, and operating procedures for the identified software minimize or eliminate the potential for breaches of system security. The developer of this design and implementation will record the security requirements in the software development plan, implement the strategy, and produce tested evidence, as part of required software products, that all security requirements have been carried out.

22. Software Risk Management

Risk Management of the CLCS process begins during the delivery definition phases.

The CLCS goal of risk management, via user community involvement, is accomplished through the User Liaison representative(s) to drive the project development process. The User Liaison, in conjunction with CLCS Project Management and System Engineering, will drive and negotiate the delivery content.

This level of communication and direction enable the project management and the project development teams to remain close and candid throughout this phase of the project. The use of the incremental reviews as part of the design panel process allows project management, through the design panel chairman, to monitor and manage project risk.

The Concept Design Panel provides the overview that indicates whether the desired delivery capability is too ambitious for a single delivery. This allows the CLCS project management to coordinate resources to provide those capabilities that not only must be completed as part of the incremental building block approach of the CLCS delivery process but also to provide the capabilities that the user community desires to have as soon as possible in the project.

The Requirements Design Panel and the Detailed Design Panel provide the CLCS project managers, through the design panel chairman, to monitor and manage/mitigate high level system level architecture issues. This insight to the product development cycle allows a great deal of risk mitigation as it is performed in 'real time', as the incremental capability is provided and the incremental system level requirements are met.

22.1 System Software Re-use

During the course of the project, the software developers will identify opportunities for obtaining software products for reuse and will evaluate the benefits and costs of these opportunities. Opportunities that provide cost benefits and are compatible with program objectives will be identified.

System software and applications developed as part of the Mission Control Center (MCC) project at the Johnson Space Center will be ported and re-used in the CLCS architecture where appropriate.

The developer will identify, evaluate, and retain, in the CM Data Base, reusable software products for use in fulfilling the requirements of the CSCI.

Representatives from the System design team determine the software reuse candidates.

23. Metrics

23.1 Metrics To Be Kept

Software Metrics and history data will be kept and maintained for the project by Systems Software, using data obtained from the Software Configuration Management Tool and provided by the CSCI Lead. These metrics include:

METRIC	Globally, by CSCI, and by CSC
Size: Source lines of code (SLOC)	√
Quality: Errors per KSLOC[1]	√
Errors found prior to CIT, (Informal Razor Issues) Includes:	√
Requirement Errors	√
Design Errors	√
Unit Test Errors	√

[1] KSLOC is 1,000 lines of source code. Source code lines are counted including comment lines, control lines, and any other complied or assembled commands.

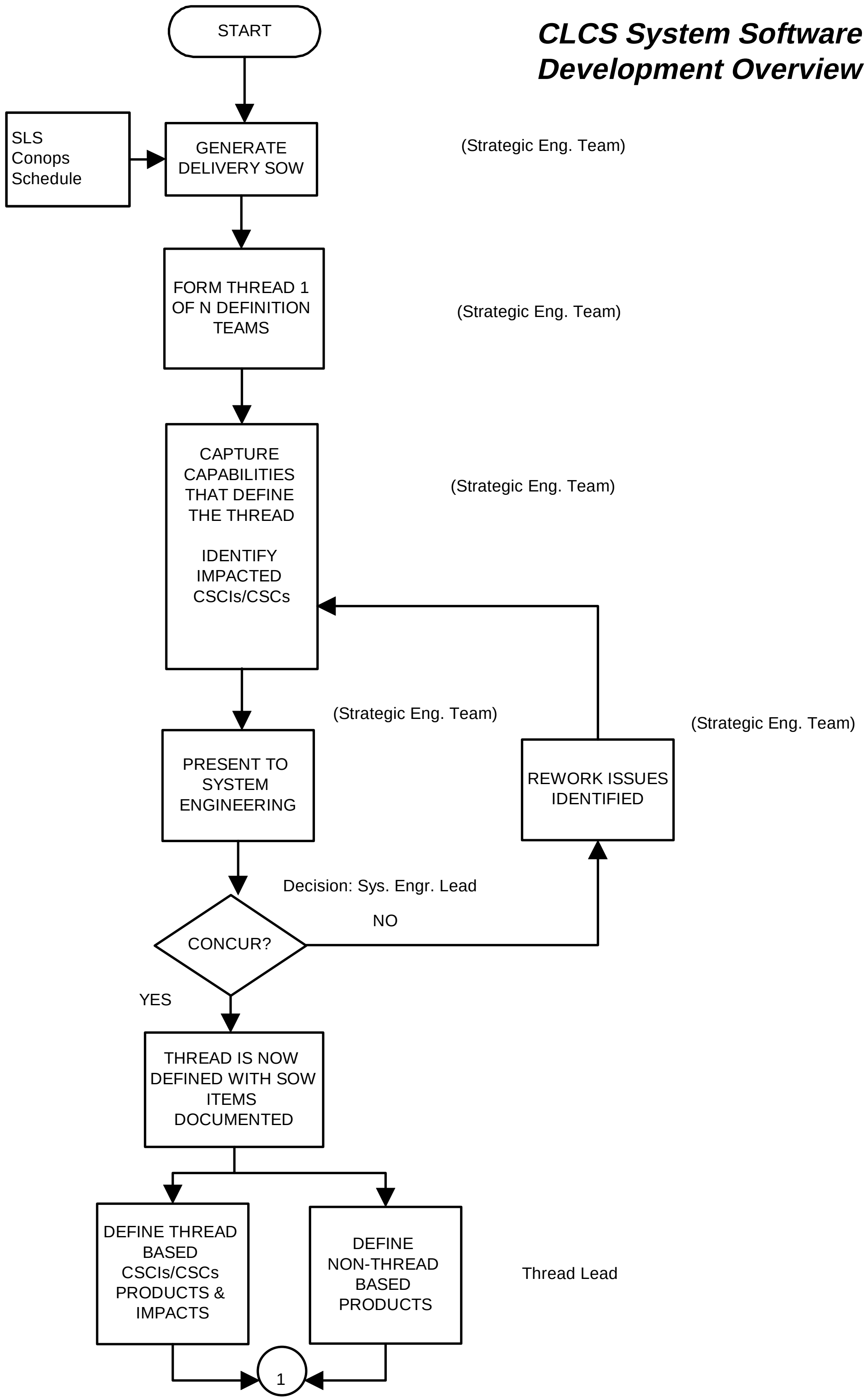
Errors found during CIT	√
Errors found during System Test	√
Post Delivery Errors	√
Productivity: KSLOC per software developer staff month	√
Description of work performed	√

Degree of complexity of work (real-time versus routine report generation) and type of work (code change v. writing new code) must also be taken into account.

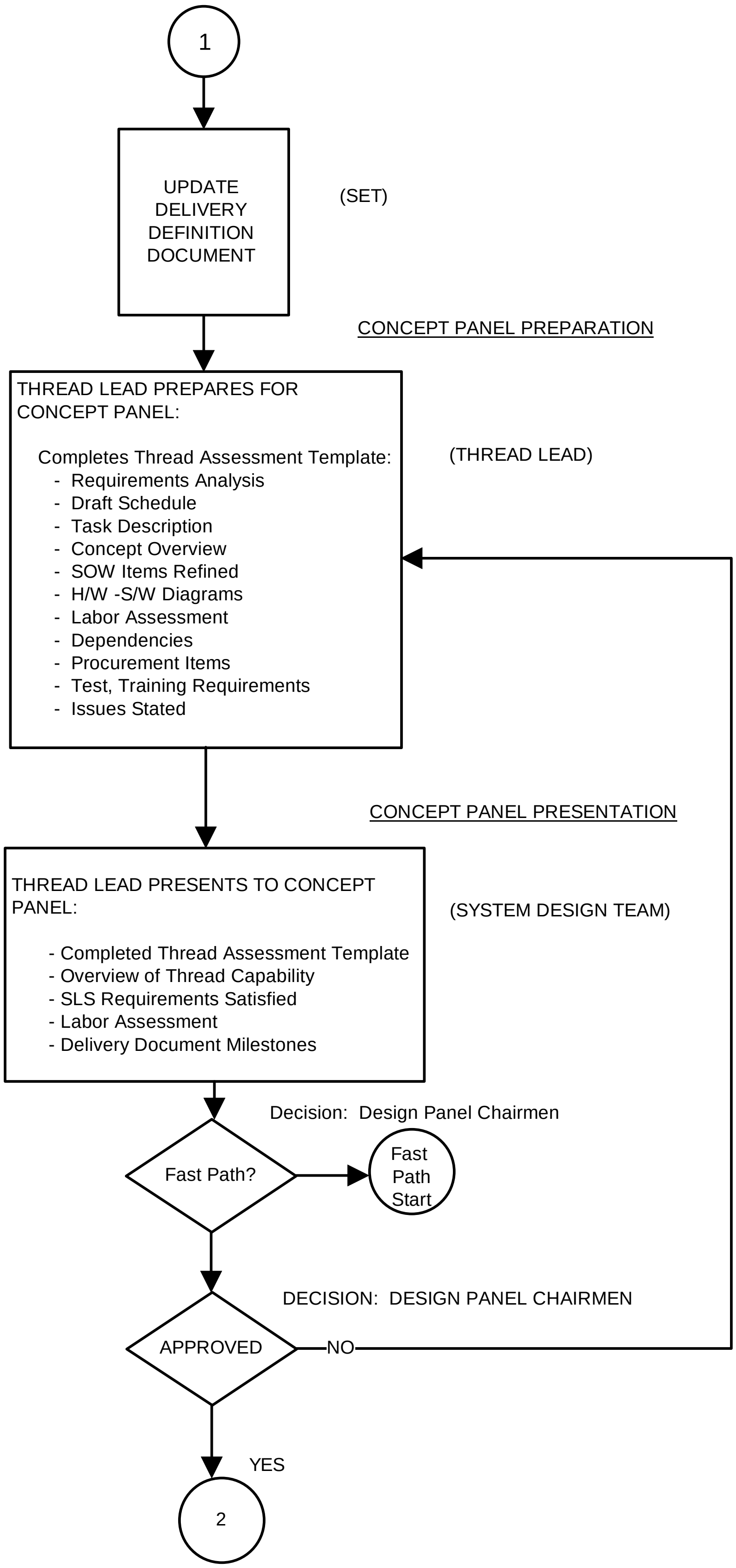
Appendix A

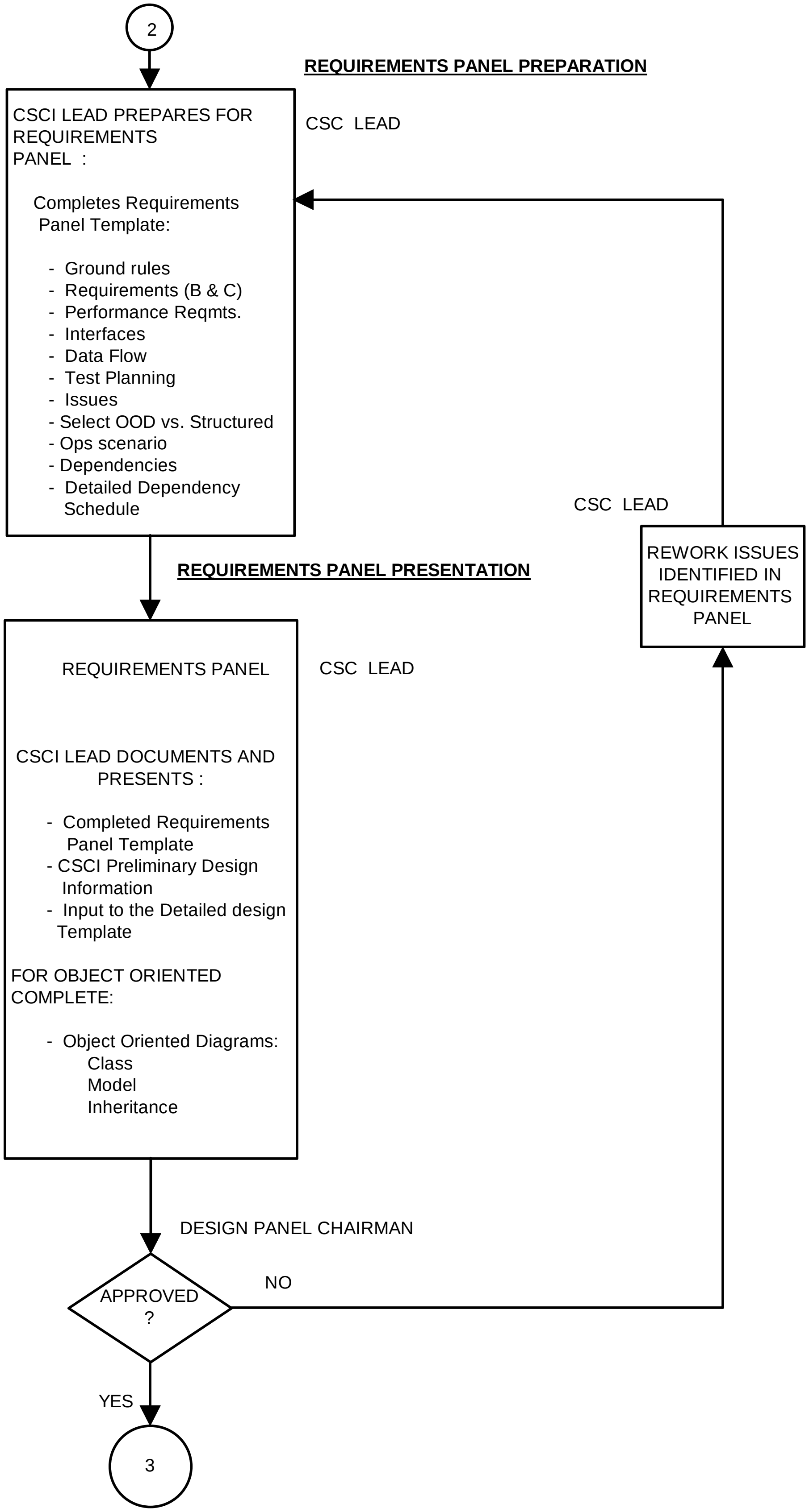
System Software Development Process

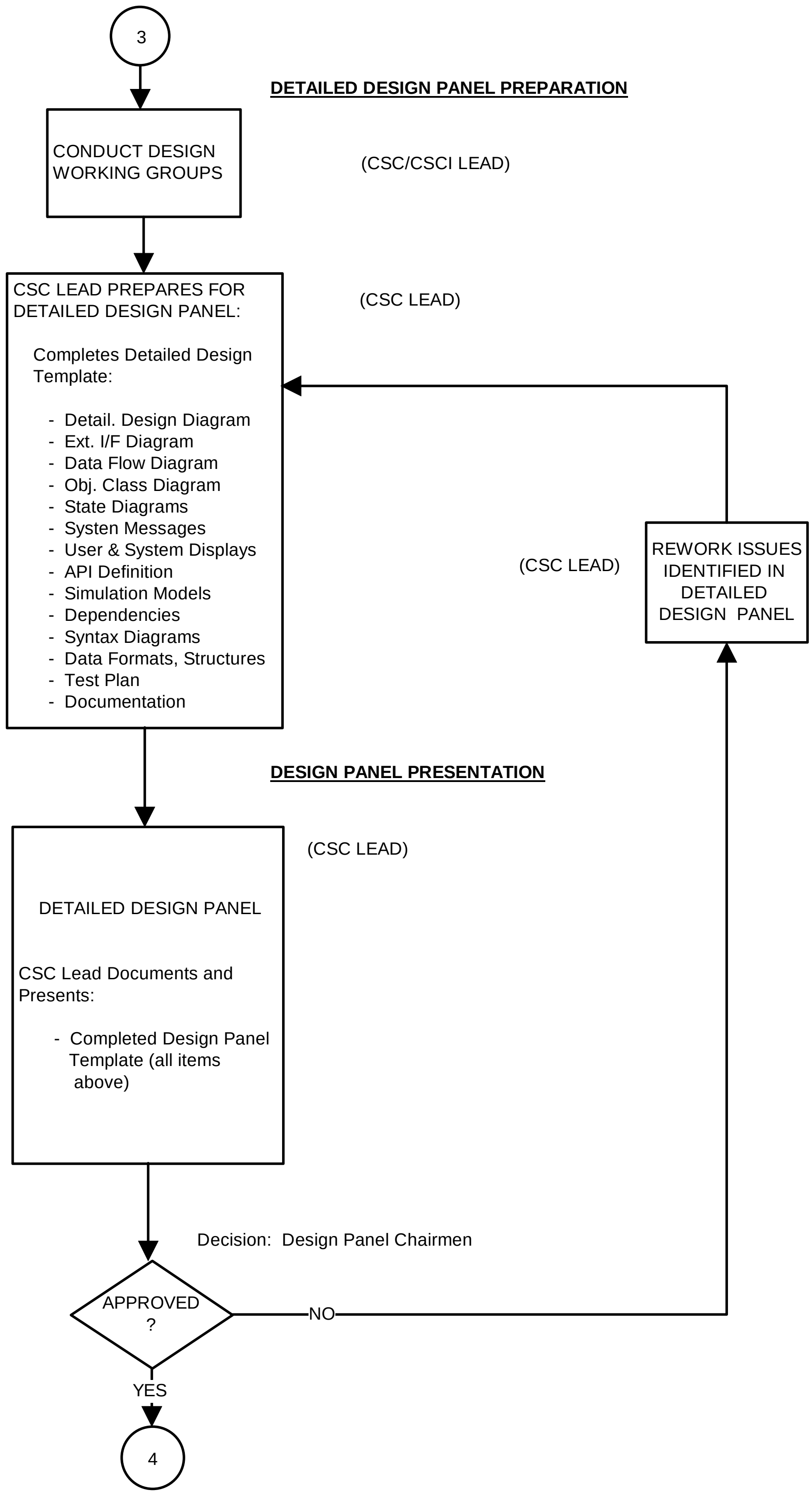
CLCS System Software Development Overview

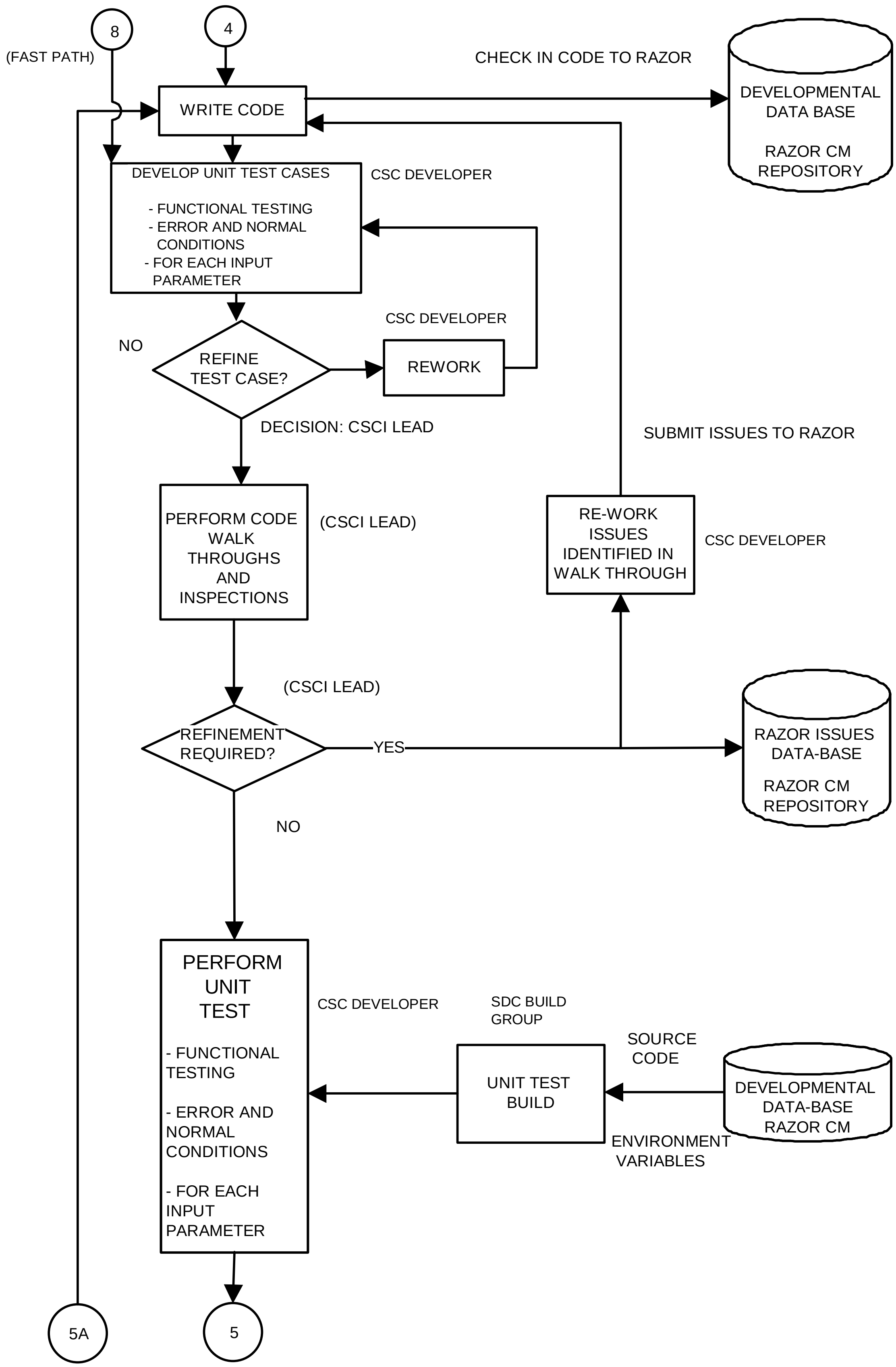


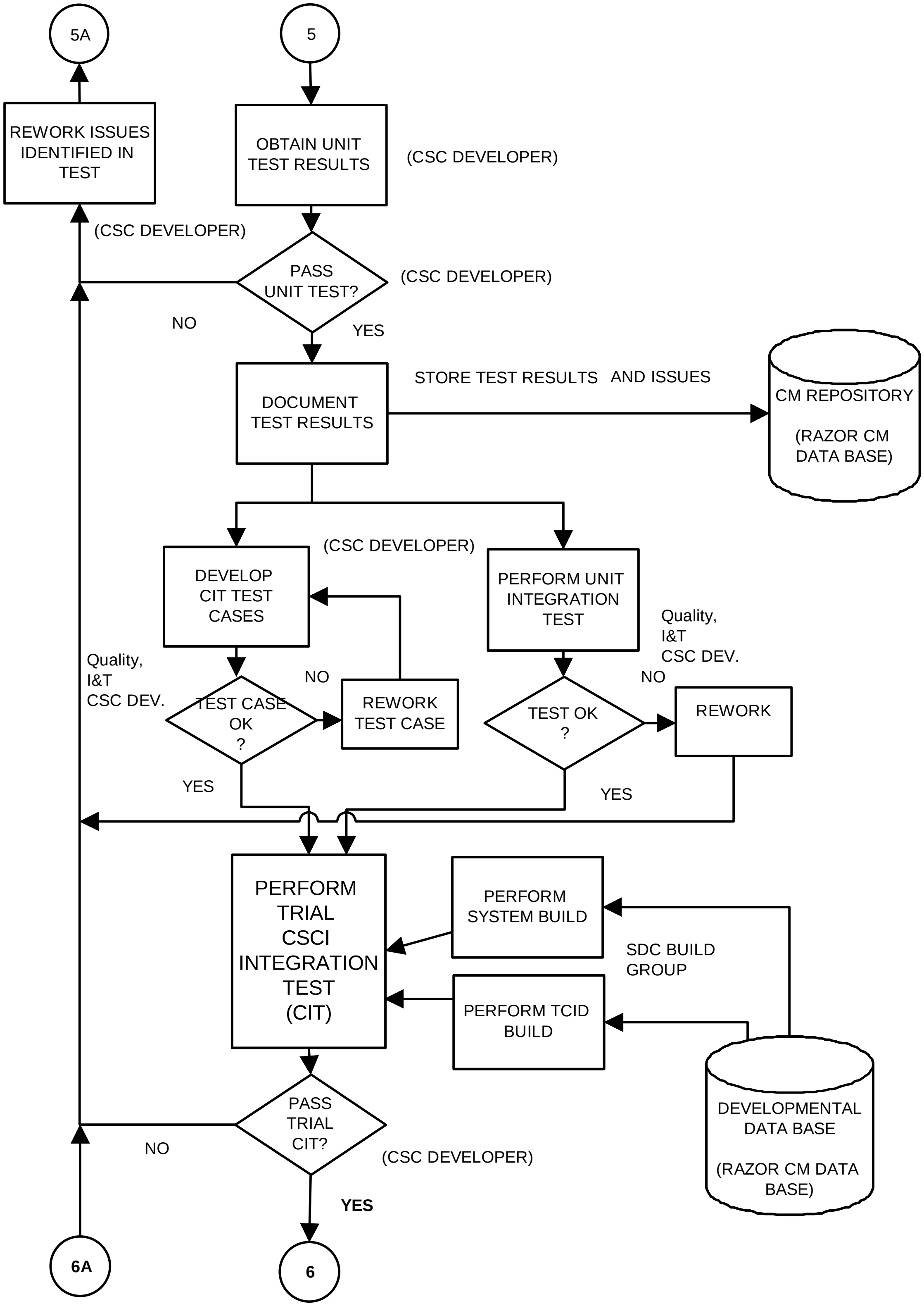
Revision 1.1
10/7/97

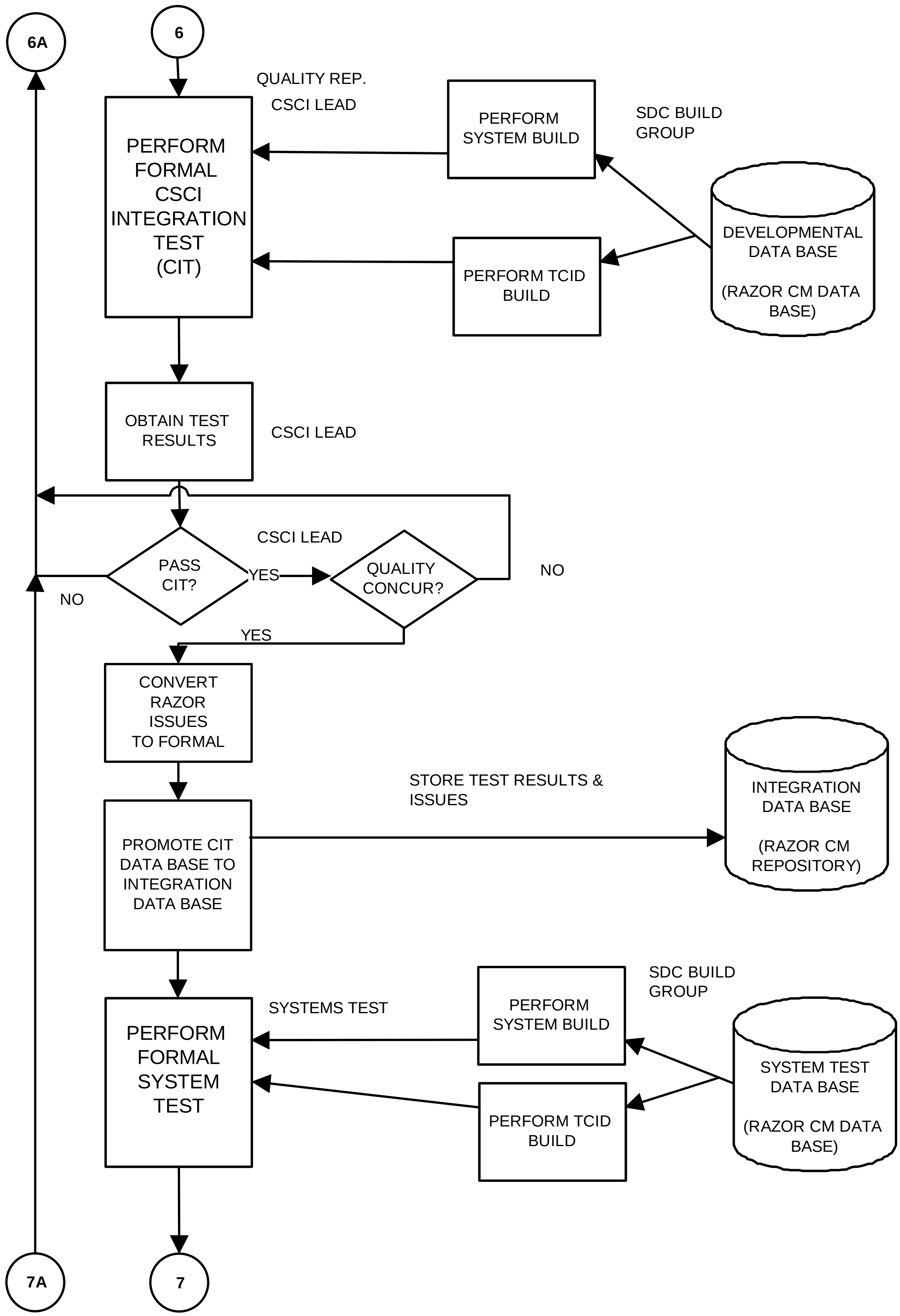

1
UPDATE DELIVERY DEFINITION DOCUMENT
(SET)
CONCEPT PANEL PREPARATION
THREAD LEAD PREPARES FOR CONCEPT PANEL:
Completes Thread Assessment Template:
- Requirements Analysis
- Draft Schedule
- Task Description
- Concept Overview
- SOW Items Refined
- H/W -S/W Diagrams
- Labor Assessment
- Dependencies
- Procurement Items
- Test, Training Requirements
- Issues Stated
(THREAD LEAD)
CONCEPT PANEL PRESENTATION
THREAD LEAD PRESENTS TO CONCEPT PANEL:
- Completed Thread Assessment Template
- Overview of Thread Capability
- SLS Requirements Satisfied
- Labor Assessment
- Delivery Document Milestones
(SYSTEM DESIGN TEAM)
Decision: Design Panel Chairmen
Fast Path?
Fast Path Start
DECISION: DESIGN PANEL CHAIRMEN
APPROVED
NO
YES
2

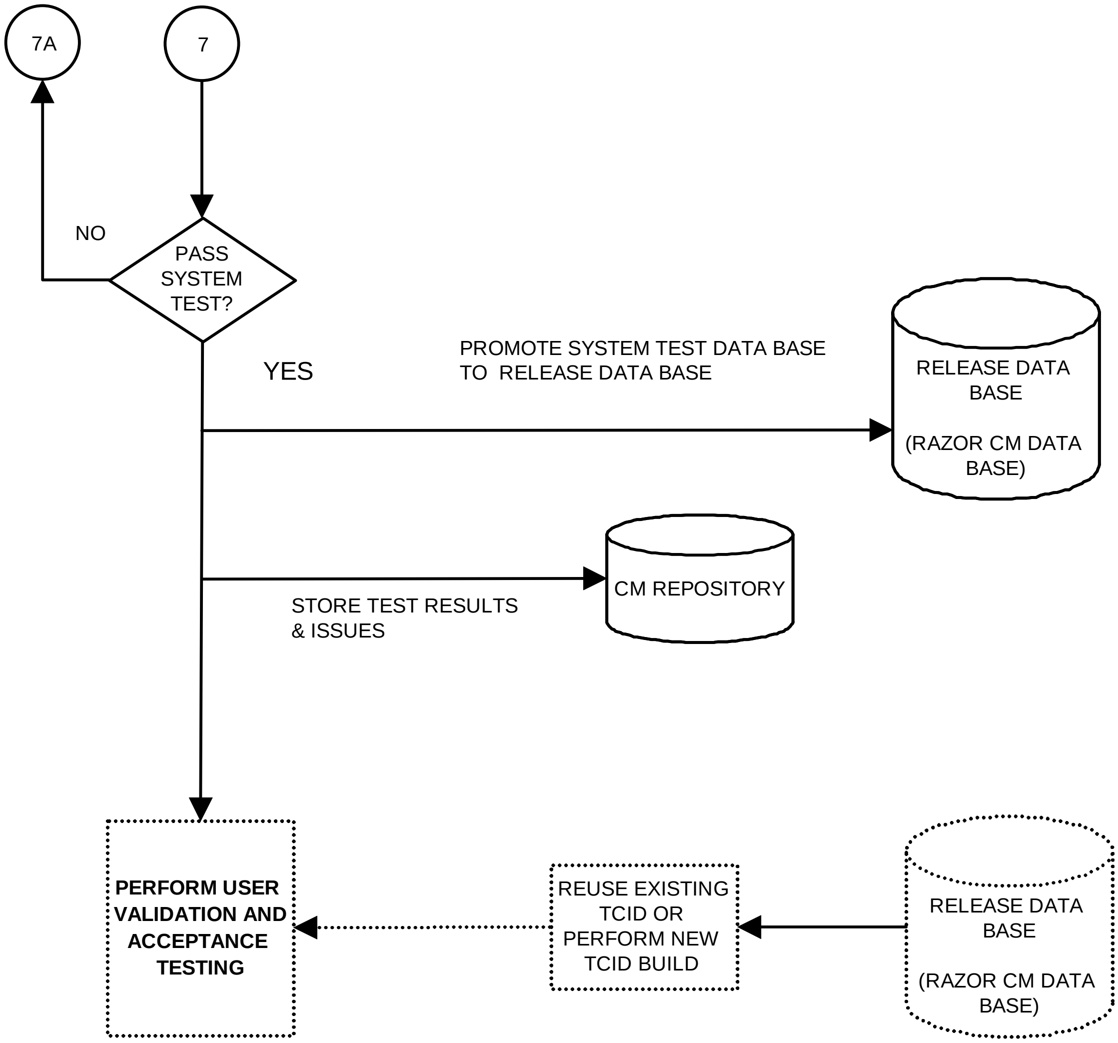

2
REQUIREMENTS PANEL PREPARATION
CSCI LEAD PREPARES FOR REQUIREMENTS PANEL :
Completes Requirements Panel Template:
- Ground rules
- Requirements (B & C)
- Performance Reqmts.
- Interfaces
- Data Flow
- Test Planning
- Issues
- Select OOD vs. Structured
- Ops scenario
- Dependencies
- Detailed Dependency Schedule
CSC LEAD
CSC LEAD
REWORK ISSUES IDENTIFIED IN REQUIREMENTS PANEL
REQUIREMENTS PANEL PRESENTATION
REQUIREMENTS PANEL
CSC LEAD
CSCI LEAD DOCUMENTS AND PRESENTS :
- Completed Requirements Panel Template
- CSCI Preliminary Design Information
- Input to the Detailed design Template
FOR OBJECT ORIENTED COMPLETE:
- Object Oriented Diagrams:
Class
Model
Inheritance
DESIGN PANEL CHAIRMAN
APPROVED ?
NO
YES
3


3
DETAILED DESIGN PANEL PREPARATION
CONDUCT DESIGN WORKING GROUPS
(CSC/CSCI LEAD)
CSC LEAD PREPARES FOR DETAILED DESIGN PANEL:
Completes Detailed Design Template:
- Detail. Design Diagram
- Ext. I/F Diagram
- Data Flow Diagram
- Obj. Class Diagram
- State Diagrams
- Systen Messages
- User & System Displays
- API Definition
- Simulation Models
- Dependencies
- Syntax Diagrams
- Data Formats, Structures
- Test Plan
- Documentation
(CSC LEAD)
REWORK ISSUES IDENTIFIED IN DETAILED DESIGN PANEL
(CSC LEAD)
DESIGN PANEL PRESENTATION
DETAILED DESIGN PANEL
CSC Lead Documents and Presents:
- Completed Design Panel Template (all items above)
(CSC LEAD)
Decision: Design Panel Chairmen
APPROVED ?
NO
YES
4


8
4
(FAST PATH)
CHECK IN CODE TO RAZOR
WRITE CODE
DEVELOPMENTAL DATA BASE
RAZOR CM REPOSITORY
DEVELOP UNIT TEST CASES
- FUNCTIONAL TESTING
- ERROR AND NORMAL CONDITIONS
- FOR EACH INPUT PARAMETER
CSC DEVELOPER
CSC DEVELOPER
NO
REFINE TEST CASE?
REWORK
DECISION: CSCI LEAD
SUBMIT ISSUES TO RAZOR
PERFORM CODE WALK THROUGHS AND INSPECTIONS
(CSCI LEAD)
RE-WORK ISSUES IDENTIFIED IN WALK THROUGH
CSC DEVELOPER
(CSCI LEAD)
REFINEMENT REQUIRED?
YES
RAZOR ISSUES DATA-BASE
RAZOR CM REPOSITORY
NO
PERFORM UNIT TEST
- FUNCTIONAL TESTING
- ERROR AND NORMAL CONDITIONS
- FOR EACH INPUT PARAMETER
CSC DEVELOPER
SDC BUILD GROUP
UNIT TEST BUILD
SOURCE CODE
DEVELOPMENTAL DATA-BASE RAZOR CM
ENVIRONMENT VARIABLES
5A
5


5A
5
REWORK ISSUES IDENTIFIED IN TEST
OBTAIN UNIT TEST RESULTS
(CSC DEVELOPER)
(CSC DEVELOPER)
PASS UNIT TEST?
(CSC DEVELOPER)
NO
YES
DOCUMENT TEST RESULTS
STORE TEST RESULTS AND ISSUES
CM REPOSITORY
(RAZOR CM DATA BASE)
(CSC DEVELOPER)
DEVELOP CIT TEST CASES
PERFORM UNIT INTEGRATION TEST
Quality, I&T CSC DEV.
Quality, I&T CSC DEV.
NO
NO
TEST CASE OK ?
REWORK TEST CASE
TEST OK ?
REWORK
YES
YES
PERFORM TRIAL CSCI INTEGRATION TEST (CIT)
PERFORM SYSTEM BUILD
SDC BUILD GROUP
PERFORM TCID BUILD
DEVELOPMENTAL DATA BASE
(RAZOR CM DATA BASE)
PASS TRIAL CIT?
NO
(CSC DEVELOPER)
YES
6A
6


6A
6
QUALITY REP.
CSCI LEAD
PERFORM FORMAL CSCI INTEGRATION TEST (CIT)
PERFORM SYSTEM BUILD
SDC BUILD GROUP
DEVELOPMENTAL DATA BASE
(RAZOR CM DATA BASE)
PERFORM TCID BUILD
OBTAIN TEST RESULTS
CSCI LEAD
CSCI LEAD
PASS CIT?
YES
QUALITY CONCUR?
NO
NO
YES
CONVERT RAZOR ISSUES TO FORMAL
STORE TEST RESULTS & ISSUES
INTEGRATION DATA BASE
(RAZOR CM REPOSITORY)
PROMOTE CIT DATA BASE TO INTEGRATION DATA BASE
SDC BUILD GROUP
SYSTEMS TEST
PERFORM SYSTEM BUILD
PERFORM FORMAL SYSTEM TEST
SYSTEM TEST DATA BASE
(RAZOR CM DATA BASE)
PERFORM TCID BUILD
7A
7


7A
7
NO
PASS
SYSTEM
TEST?
YES
PROMOTE SYSTEM TEST DATA BASE
TO RELEASE DATA BASE
RELEASE DATA
BASE
(RAZOR CM DATA
BASE)
CM REPOSITORY
STORE TEST RESULTS
& ISSUES
PERFORM USER
VALIDATION AND
ACCEPTANCE
TESTING
REUSE EXISTING
TCID OR
PERFORM NEW
TCID BUILD
RELEASE DATA
BASE
(RAZOR CM DATA
BASE)

Fast Path Development Overview